Artificial Intelligence
(Problem Solving through Search Algorithms)

Prof K R Chowdhary

CSE Dept., MBM University

February 3, 2025

Lecture #3

Depth-First Search (DFS)

To perform a DFS search, we generate all the next states for the root node, then pickup the left-most node, generate the children for this, check for goal, and repeat this, until we reach to goal or the dead end.

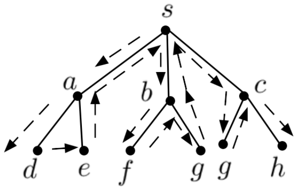


Figure 1: Depth-first Search.

Algorithm 1 DFS(Input: **G**, **S**, **Goal**)

List = [S]
repeat
  if **List**.*Head* = **Goal** **then**
    return *success*
  end if
  generate children set **C** of **List**.*Head*
  delete **List**.*Head*
  insert **C** at begin of **List**
until **List** = []
return *fail*

Analysis of BFS, DFS

- Consider the figure 5 in Slide 2, and Fig. 1 in this slide. Assume that the goal node is e. Using the *DFS* it needs three steps to reach to node e. Five steps if *BFS* is used.
- So, for a deep goal, *DFS* is better. If to be searched was c, *BFS* required three steps, and *DFS* needs total 7 steps to reach goal.
- Thus, which approach is best, depends on the position of goal node.
- Also, if branching factor b (number of branches per node) is large, the *DFS* is better suited, and *BFS* is worst.
- Efficiency of search depends on the structure of tree as well as search method used. Worst case complexity for n nodes is $1 + b + b^2 + \cdots + b^n = O(b^d)$. (d=depth of tree).

Towers of Hanoi Problem: Move all the disks from tower A to tower B, using intermediate tower I, and at no time any tower have bigger disk above a smaller disk in stack.

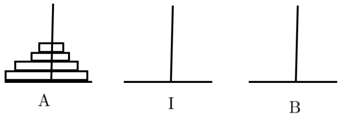


Figure 2: Towers of Hanoi Problem, with initial condition.

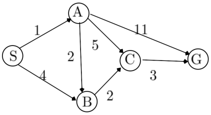
Prolog Code:

```
move(A,B):- nl,
 write('move top from '),
 write(A),
 write(' to '),
 write(B).

transfer(1,A,B,I) :- move(A,B).
transfer(N, A, B, I):- N > 1,
     M is N -1,
     transfer(M, A, I, B),
     move(A, B),
     transfer(M, I, B, A).
```

A^* Search: Informed Search

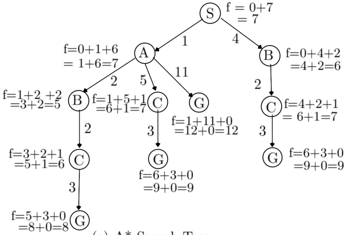
- In A^* algo., cost of a path is sum of costs of its arcs. A^* employs an additive evaluation function $f(n) = g(n) + h(n)$, where $g(n)$ is the cost of the currently evaluated path from s to n and h is a heuristic estimate of cost of path between node n and some goal node [2].
- $g(n)$ is *order preserving* and $h(n)$ depends only on the description of the node n, so $f(n)$ is also order preserving.



(a) Graph.

State	h
S	7
A	6
B	2
C	1
G	0

(b) Heuristic Table.



(c) A*-Search Tree.

Figure 3: Graph and A*-Search-tree.

A^* Search

- The figure 3 shows graph, heuristic function table for $h(n)$ for the graph, and tree constructed for A^* search, for start state S and goal state G.
- To expand the next state (node), the one having smallest value of function f is chosen. Function f for node n is sum of : g value of the parent of n, distance from parent of n to n, and heuristic value (estimated distance from n to goal) indicated by h.
- Based on the criteria set for A^* (i.e., to always expand the node having smallest value of f), the order of nodes expanded for figure 3(a), and shown in search-tree figure 3(c) with start node S and goal node G are: $(G,0)$, $(B,6)$, $(A,7)$, $(B,5)$, $(C,6)$,$(C,7)$ (with parent A), $(C,7)$ (with parent B), $(G,8)$, $(G,9)$, $(G,12)$.
- Finally, we note that the best path is corresponding to goal $(G,8)$, and it is: S, A, B, C, G.

Say, N = 5. "transfer(M, A, I, B) means transfer 4 disks from A to I using B as intermediate disk. Then move last disk from A to B (move (A, B)). Then transfer 4 disks from I to B, using A as intermediate (transfer(M, I, B, A)). Uses recursion.

Real-world applications of search algorithms

Path finding in Robotics:

- *Application*: Robots often use search algorithms (e.g., A*) to navigate a physical environment and avoid obstacles.
- *Example*: Autonomous robots using search algorithms to find the shortest path in an indoor map or warehouse See (program bfs_robot.py).
- *Resource Material*: ROS (Robot Operating System) or Gazebo for real-world simulations.

Electric grid's Load Distribution:

- *Application*: Search algorithms can be used to optimize load distribution across power plants/grids to minimize cost or maximize efficiency.
- *Example*: Search algorithms that find most efficient way to distribute electricity from power plants to cities based on demand (program: gridload.py).
- *Resource Material*: Power grid optimization problems, e.g., *load flow analysis*, where search algorithms can model the problem space.

Real-world applications of search algorithms and References

Structural Optimization in Civil Engineering:

- *Application*: Search algos. used to design structures with minimum material usage while meeting certain safety requirements.
- *Example*: Finding most optimal truss design or bridge layout w.r.t weight and cost constraints.
- *Resource Material*: Genetic algorithms, simulated annealing, etc.

[1] Chowdhary, K.R. (2020). State Space Search. In: Fundamentals of Artificial Intelligence. Springer, New Delhi. https://doi.org/10.1007/978-81-322-3972-7_8(pages: 217-237)

[2] Chowdhary, K.R. (2020). Heuristic Search. In: Fundamentals of Artificial Intelligence. Springer, New Delhi. https://doi.org/10.1007/978-81-322-3972-7_9